The Minimal Cost Algorithm for off-line Diagnosability of Discrete Event Systems

Zhujun Fan

Department of Mathematics, Zhongshan University, Guangzhou 510275, China

Abstract

The failure diagnosis for *discrete event systems* (DESs) has been given considerable attention in recent years. Both on-line and off-line diagnostics in the framework of DESs was first considered by Lin Feng in 1994, and particularly an algorithm for diagnosability of DESs was presented. Motivated by some existing problems to be overcome in previous work, in this paper, we investigate the minimal cost algorithm for diagnosability of DESs. More specifically: (i) we give a generic method for judging a system's off-line diagnosability, and the complexity of this algorithm is polynomial-time; (ii) and in particular, we present an algorithm of how to search for the minimal set in all observable event sets, whereas the previous algorithm may find *non-minimal* one.

Index Terms

Discrete event systems, observable event sets, failure detection, fault diagnosis, minimal cost algorithm.

I. Introduction

WITH man-made systems becoming more and more complex, detecting and locating component failure is not a simple task. Therefore, there is a strong need for a systematic study of diagnostic problems and diagnosability issues [30]. As an important kind of man-made systems, discrete event system (DES) is a dynamical system whose state space is discrete and whose states can only change as a result of asynchronous occurrence of instantaneous events over time [2]. Up to now, DESs have been successfully applied to provide a formal treatment of many technological and engineering systems [3, 5, 16]. Naturally, the diagnosability of DESs is of theoretical and practical importance.

Actually, diagnosability of DESs has received extensive attention in recent years (for example, [1, 4, 6-9, 11-15, 17-32]). Especially, in [15], the definitions of "off-line" diagnosability and "on-line" diagnosability were introduced, and both "off-line" diagnostic algorithm and "on-line" diagnostic algorithm were

significantly established in the framework of DESs. However, the algorithms presented in [15] have some shortcomings: 1) the computational complexity of "off-line" diagnostic algorithm is exponential in general; 2) and "off-line" diagnostic algorithm could not find the minimal one in observable events sets (OESs), and the algorithm of how to inspect an automaton being diagnosable was not yet given. Motivated by these issues, our goal in this paper is to solve these problems.

The remainder of the paper is organized as follows. In Section II, we first introduce a general framework of diagnosability of DESs, and then explain the lost of Algorithm 1 in [15]. In Section III, the definition of "off-line" diagnostics is first provided, and we then present a polynomial-time algorithm to realize it. In Section IV, we demonstrate the principle of finding the minimal set in an OES, and particularly, present our new algorithm to realize it; Section V provides two examples to illustrate these algorithms in Sections III and IV. Finally some remarks are made in Section VI to conclude the paper.

II. Preliminaries

A. A general framework for automata and diagnostics

1) $DFAs$: A deterministic finite automaton (DFA) can be formally defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the starting state, and $F \subseteq Q$ is a set of accepting states. Operation of the DFA begins at q_0, and movement from state to state is governed by the transition function δ. δ must be defined for every possible state in Q and every possible symbol in Σ.

A DFA can be represented visually as a directed graph. Circular vertices denote states, and the set of directed edges, labeled by symbols in Σ, denotes δ. The transition function takes the first symbol of the input string, and after the transition this first symbol is removed. If the input string is ϵ (the empty string), then the operation of the DFA is halted. If the final state when the DFA halts is in F, then the DFA can be said to have accepted the input string it was originally given. The starting state q_0 is usually denoted by an arrow pointing to it that points from no other vertex. States in F are usually denoted by double circles.

$DFAs$ recognizes regular languages, and can be used to test whether any string is in the language it recognizes. As it is known, DFAs have been used to model DESs [2]. In the following, we use DFA to represent a DES.

2) Diagnosability of Discrete Event Systems: We model the system to be diagnosed as a pair $G = (M, \Sigma_c)$. The first component M denotes a nondeterministic Mealy automaton:

$$M = (\Sigma, Q, Y, \delta, h)$$

where Σ is the set of finite events; Q is the set of finite states; Y is the output alphabet space; $\delta : \Sigma \times Q \rightarrow 2^Q$ is the state transition function. $\delta(\sigma, q)$ gives the set of possible next states if σ occurs at q; and $h : \Sigma \times Q \rightarrow Y$ is the output function, $h(\sigma, q)$ is the observed output when σ occurs at q. The second component $\Sigma_c \subseteq \Sigma$ is the set of controllable events, where the controllability of events is interpreted in a strong sense: a controllable event can be made to occur if physically possible.

States of the system describe the conditions of its components. Therefore, to diagnose a failure is to identify which state or set of states the system belongs to. Thus, depending on the requirements on diagnostics, we partition the state space Q into disjoint subset (cells) and denote the desired partition by T. The state in the same cell are viewed as equivalent as far as failures under consideration are concerned. The model is rather general since we do not put any restrictions on T.

3) Some notations: For convenience, we give some notations. Let

$$M = \{\sigma_1, \sigma_2, \ldots, \sigma_n\},$$

where $\sigma_i, i = 1, \ldots, n$, are the observed events, and the cost of M is denoted by

$$C(M) = \{c(\sigma_1), c(\sigma_2), \ldots, c(\sigma_n)\}.$$

where $c(\sigma_i)$ means the cost of observes event $\sigma_i, i = 1, \ldots, n$, respectively. Without loss of generality, we suppose that

$$c(\sigma_1) \geq c(\sigma_2) \geq \ldots \geq c(\sigma_n).$$

Let $\Sigma_o \in OES$. We denote $C(\Sigma_o) = \sum\limits_{\sigma_i \in \Sigma_o} c(\sigma_i)$, which presents the cost of Σ_o, and

$$minL(\Sigma_o) = min\{i : \sigma_i \in \Sigma_o\}.$$

An important problem is how to find the smallest observable event set that makes G diagnosable for a given partition T. In order to solve this problem, we define the set of all observable event sets (OESs) that ensure the diagnosability of the system as:

$OES(T)$ = $\{\Sigma_o \subseteq \Sigma$: G is diagnosable with respect to Σ_o and $T\}$.

B. Lost of Algorithm 1 of [15]

1) Algorithm 1 of [15]: In order to remove events one by one in the given order until the diagnosability of the system is no longer ensured, Algorithm 2.1 (Fig. 1) was presented in [15].

However, Algorithm 2.1 has some shortcomings, we will illustrated them in next subsection.

Algorithm 2.1: (minOES)

Input: Read $G = (M, \Sigma_C)$, $M = (\Sigma, Q, Y, \delta, h)$,$T$;

Initialization:$minOES$:= Σ;

Removal: **For** $i = 1$ to n **do**

begin $minOES := minOES \setminus \{\sigma_i\}$;

if G is not diagnosable with respect to $minOES$ an T **then**

$minOES := minOES \bigcup \{\sigma_i\}$;

end;

Output: Return $minOES$;

Fig. 1. Algorithm 1 of [15].

2) Lost Example of Algorithm 1 of [15]: In fact, the "off-line" diagnostic algorithm (Algorithm 2.1) could not find the minimal one in observable events sets. For example, let

$$M = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\},$$

and the cost of M is

$$C(M) = \{13, 9, 7, 5, 2\}.$$

$OES(T) = \{E_1\} \cup \{E_2\} \cup \{\Sigma \subseteq M : E_1 \subseteq \Sigma, or, E_2 \subseteq \Sigma\}$, where $E_1 = \{\sigma_2, \sigma_5\}$, and $E_2 = \{\sigma_3, \sigma_4, \sigma_5\}$.

G is diagnosable with respect to given T and an element of $OES(T)$.

If we use Algorithm 2.1, we can get the $minOES = E_2$, the cost of E_2 is $7+5+2 = 14$. However, the cost of E_1 is $9+2 = 11$, which is less than the cost of E_2. Therefore, the $minOES$ is not the minimal cost of $OES(T)$.

III. Off-line Diagnostics

Off-line diagnostics means that diagnosis is performed when the system is not in normal operation [15]. For example, what a mechanic does to an automobile in a repair shop can be viewed as off-line diagnostics. In order to perform off-line diagnostics, one can "open" the system, access the inside, do various tests, and measure responses that may not be available from the system outputs. In fact, during off-line diagnostics, the system is not actually in operation. Therefore, the failure status of system components will not change, unless such changes are made in purpose. So tests can be designed with great flexibility and the order of testing is not critical as far as diagnosability is concerned.

A. Off-line Diagnostics [15]

For off-line diagnostics, we specialize the model introduced in the previous section by assuming that the output are events observed. That is, $Y = \Sigma_o$, where $\Sigma_o \subseteq \Sigma$ is the set of observable events and the output map $h : \Sigma \times Q \to \Sigma_o$ is a projection defined as:

$$h(\sigma, q) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_o, \\ \epsilon & \text{otherwise}, \end{cases}$$

where ϵ is the empty string.

As it was discussed before, in off-line diagnostics all events are assumed to be controllable. Therefore, $\Sigma_c = \Sigma$. Since the failure states of system components will not change, information derived from all the test outputs are updated and relevant.

During off-line diagnostics, if an event $\sigma \in \Sigma_o$ is observed, then the possible state of the system is:

$$Q(\sigma) = \{q \in Q : (\exists q' \in Q)\delta(\sigma, q') = q\}. \tag{1}$$

Hence, we know every state of the system is in either $Q(\sigma)$ or $Q - Q(\sigma)$ after observing σ. That is, each observable event partitions the state space into:

$$T_\sigma = \{Q(\sigma), Q - Q(\sigma)\}. \tag{2}$$

Since there is not restriction on the tests performed in off-line diagnostics, we can observe all observable events that are physically possible and then determine which states the system is in. If this information is sufficient for us to determine which component is broken (i.e., which cell of T the system is in), then we say the system is off-line diagnosable. Formally:

Definition 3.1: G is said to be off-line diagnosable with respect to T if

$$\bigwedge_{\sigma \in \Sigma_o} T_\sigma \leq T \tag{3}$$

where $\wedge$ denotes conjunction and $\leq$ means "is finer than".

Clearly, diagnosability depends on both the observable event set Σ_o and the desired partition T.

B. An algorithm for off-line diagnosability

In Section III, we have introduced the definition of "off-line" diagnostics (see Definition 3.1 and equation (3) in Section III). In equation (3), the right part T is given by the system. Now we must first calculate the left part $\bigwedge_{\sigma \in \Sigma_o} T_\sigma$, where T_σ is given in equation (2) of Section III and $Q(\sigma)$ is given in equation (1) of Section III. From these two equations, given an element σ, for every element of Q, it must be in $Q(\sigma)$ or not in $Q(\sigma)$ (in $Q \setminus Q(\sigma)$). So we can use one bit to identify every element of Q in $Q(\sigma)$ or not in $Q(\sigma)$ (i.e, 1 for elements in $Q(\sigma)$ and 0 for elements not in $Q(\sigma)$). Now we give algorithms to realize them.

Algorithm 3.2: (QC).

Input: δ, σ, Q, q;

Initialization: Set $m = |Q|, QC := False$;

Judge: **for** $i = 1$ to m **do**

if $\delta(\sigma, q_i) == q$ **then**

$QC := True$,**break**;

Output: Return QC;

Fig. 2. Algorithm:whether a state q in $Q(\sigma)$ or not, that is $\delta(\sigma, Q) = q$ or $\delta(\sigma, Q) \neq q$.

Algorithm 3.3: (TQC).

Input: δ, Σ_o, Q;

Initialization: Set $m = |Q|, n = |\Sigma_o|, s_j = 0(j = 1..m)$

Intersection: **for** $i = 1$ to n **do**

for $j = 1$ to m **do**

if $\delta(\sigma_i, Q) = q_j$(Algorithm 3.2) **then**

$s_j| = (1 << (i-1))$;

Output: Return $s_j(j = 1..m)$;

Fig. 3. Algorithm: $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$.

1) Algorithm: Algorithm 3.2 (Fig. 2) gives whether a state q in $Q(\sigma)$ or not, that is $\delta(\sigma, Q) = q$ or $\delta(\sigma, Q) \neq q$.

Algorithm 3.3 (Fig. 3) gives the calculation of $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$.

In Algorithm 3.3, all the elements of F ($F \in \bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$) have the same value s_j, since they have the same operation in Algorithm 3.3. And $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma \leq T$ means that every element of $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$ is the subset of G ($G \in T$). The reverse proposition means that there exists an element of $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$, not all of its elements are the elements of $G(G \in T)$. From this, we have Algorithm 3.4 (Fig. 4).

2) Algorithm Complexity: In Algorithm 3.2, in “Judge” recycle, the bad time is m. So the time complexity of Algorithm 3.2 is $O(m)$.

In Algorithm 3.3, in “Intersection” recycle, it has two loops, the complexity of first line is $O(n)$; the complexity of second line is $O(m)$. In third line, it calls the Algorithm 3.2, so the bad time is $O(m)$; and then the total complexity in “Intersection” recycle is $O(m^2n)$. Therefore, the time complexity of Algorithm

Algorithm 3.4: (OFD).

Input: δ, Σ_o, Q, T;

Initialization: Set $OFD := True$;

Diagnosing: Get $s_j(j = 1..m)$ from Algorithm 3.3;

Applied Quicksort Algorithm to $s_j(j = 1..m)$;

for $j = 1$ to $m - 1$ **do**

if $(s_j == s_{j+1})$ **then**

begin Find $T_i \in T$, s.t.$\sigma_j \in T_i$;

if $\sigma_{j+1} \overline{\in} T_i$ **then**

$OFD := False$;

end;

Output: Return OFD;

Fig. 4. Algorithm: $\bigwedge_{\sigma \in \Sigma_o} T_\sigma \leq T$.

3.3 is $O(m^2 n)$.

In Algorithm 3.4, in "Diagnosing" recycle, it first calls the Algorithm 3.3, the time complexity is $O(m^2 n)$; then it calls the Quicksort Algorithm, the bad time complexity is $O(m^2)$; for the other lines, it has one loop, the total complexity is $O(m^2)$. In conclusion, the time complexity of Algorithm 3.4 is $O(m^2 n)$.

IV. New Algorithm for finding the minimal one in $OESs$

A. *Finding the minimal one in $OESs$*

We would like to find a minimal element in $OES(T)$ as follows.

Proposition 4.1: If $OES(T)$ is not null, then the minimal elements of $OES(T)$ exist, but may not be unique.

Proof: The proof of the existence of minimal elements is straightforward. Since Σ is finite, 2^Σ is a finite set. Notice that $\Sigma_o \subseteq \Sigma$, therefore, Σ_o is an element of 2^Σ, and the elements of $OES(T)$ are finite. As a result, there exists a minimal element in $OES(T)$.

The following example shows that the minimal elements of $OES(T)$ may not be unique. Let

$$\begin{aligned} \Sigma &= \{\alpha, \beta, \gamma\} \\ Q &= \{q_1, q_2\} \\ \delta(\alpha, q_1) &= \{q_2\} \\ \delta(\beta, q_2) &= \{q_1\} \\ \delta(\gamma, q_1) &= \{q_2\} \\ \delta(\sigma, q) &= \emptyset \ \ otherwise, \end{aligned}$$

and

$$T = \{\{q_1\}, \{q_2\}\}.$$

Obviously, $\{\alpha\}$, $\{\beta\}$ and $\{\gamma\}$ are minimal elements of $OES(T)$. ■

From Proposition 4.1 in Section III, we conclude that we may be able to find more than one set of observable events, and each set is minimal in the sense that removing any event from the set will make the system not diagnosable. Practically, we can find a cost-effective minimal observable event set by first ordering the events in terms of the difficulty (and hence cost) in detection. This directly gives the Algorithm1 of [15](Fig. 1 Algorithm 2.1).

B. New Algorithm

From the Example in Section II-B.2, we know that the $minOES$ given by Algorithm 2.1 is not the minimal cost one. Therefore, we will modify Algorithm 2.1 to find the minimal cost one in this subsection.

Proposition 4.2: By Algorithm 2.1, we get the $minOES$, whose cost is $C(minOES)$ and whose minimal label is $minL(minOES)$. If there exists an $\Sigma_o \in OES(T)$, with $C(\Sigma_o) < C(minOES)$, then we have

$$minL(\Sigma_o) \leq minL(minOES).$$

Proof: If $minL(\Sigma_o) > minL(minOES)$. Set $L = minL(minOES)$ is the minimal index of set $minOES$. In Algorithm 2.1, when $I = L$, G is diagnosable with respect to $minOES$ and T, and the next step of Algorithm 2.1 is not executed. So $\sigma_L \overline{\in} minOES$, and then $L \neq minL(minOES)$. Consequently, $minL(\Sigma_o) \leq minL(minOES)$. ■

Now we present a new algorithm 4.3 (Fig. 5) to find the minimal cost one.

C. Necessary Element

Definition 4.4: (Necessary Element) Suppose $\Sigma \in OES(T)$. If $\sigma_i \in \Sigma$, but $\Sigma \setminus \{\sigma_i\} \overline{\in} OES(T)$, then we call σ_i necessary element with respect to T.

Algorithm 4.3: (MMOES).

Input: $G = (M, \Sigma_C)$, $M = (\Sigma, Q, Y, \delta, h)$,T; the order $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$; the cost of $\Sigma, C(\Sigma) = \{c(\sigma_1), c(\sigma_2), \ldots, c(\sigma_n)\}$;

Initialization: Get $minOES$ by Algorithm 2.1;

Set $lmS = minL(minOES)$, $cmS = C(minOES)$;

Set $H = \{\Sigma_o \subseteq \Sigma : minL(\Sigma_o) \geq lmS, C(\Sigma_o) < cmS\}$;

Set $ng = |H|$, and $H = \{H_1, H_2, \ldots, H_{ng}\}$;

Testing diagnosability: for $i = 1$ to ng **do**

begin if (G is diagnosable with respect to H_i an T) AND ($C(H_i) < cmS$) **then**

$minOES = H_i, cmS = C(H_i)$;

end;

Output: Return $minOES$;

Fig. 5. Algorithm:Modify $MinOES$.

Proposition 4.5: If $\Sigma_o \in OES(T)$, and $\Sigma_o \subseteq F$, then $F \in OES(T)$.

Proof: Because

$$\bigwedge_{\sigma \in F} T_\sigma \leq \bigwedge_{\sigma \in \Sigma_o} T_\sigma \leq T,$$

the proposition holds true. ■

Proposition 4.6: If σ_i is a necessary element, then for any $\Sigma_o \in OES(T)$, $\sigma_i \in \Sigma_o$.

Proof: (proof by contradiction) If the theorem is not true, there exists an $\Sigma_o \in OES(T)$, with $\sigma_i \overline{\in} \Sigma_o$. Therefore, $\Sigma_o \subseteq \Sigma \setminus \{\sigma_i\}$. And σ_i is a necessary element, $\Sigma \setminus \{\sigma_i\} \overline{\in} OES(T)$. So $\Sigma_o \overline{\in} OES(T)$, which is a contradiction to assumption. So the proposition is true. ■

Definition 4.7: (Necessary element set) $NES(T)$={ σ_i : σ_i is necessary element with respect to $T\}$.

Corollary 4.8: For any $\Sigma_o \in OES(T)$, $NES(T) \subseteq \Sigma_o$.

Algorithm 4.9: (NES).

Input: $G = (M, \Sigma_C)$, $M = (\Sigma, Q, Y, \delta, h)$,T; the order $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$;

Initialization: Set $NES := \emptyset$;

AddElement: **for** $i = 1$ to n **do**

begin **if** G is not diagnosable with respect to $\Sigma \setminus \{\sigma_j\}$ and T **then**

$NES := NES \cup \{\sigma_j\}$;

end;

Output: Return NES;

Fig. 6. Algorithm:NES(T).

Proof: For any $\Sigma_o \in OES(T)$, and any $\sigma_i \in NES(T)$, there exists $\sigma_i \in \Sigma_o$ (see Proposition 4.6). So $NES(T) \subseteq \Sigma_o$. ■

We introduce $NES(T)$ to reduce the computing time. We partition the finite events space Σ into two disjoint subsets $NES(T)$ and $\Sigma \setminus NES(T)$. The set $NES(T)$ must include all the elements in $OES(T)$. If we get $NES(T)$ first, Algorithm 4.3 in this section need only compute in set $\Sigma \setminus NES(T)$. This may reduce computing complexity.

D. Algorithm Complexity

Suppose the time (of whether G is not diagnosable with respect to $minOES$ and T) is T_G, where $T_G = O(m^2 n)$.

In Algorithm 2.1, in "Removal" recycle, the bad time is $n \times T_G$, and the time-complexity of Algorithm 2.1 is $O(m^2 n^2)$.

In Algorithm 4.3,in "Initialization" recycle, it first calls Algorithm 2.1 to get $minOES$, the bad time is $O(m^2 n^2)$; and then it get set H, its a 0-1 pack problem, the bad time is $O(n \times cmS)$; in "Testing diagnosability" recycle, the bad time is $ng \times T_G$; therefore the time-complexity of Algorithm 4.3 is $O(m^2 \times n \times ng)$.

In Algorithm 4.9, in "AddElement" recycle, the bad time is $n \times T_G$, so the time-complexity of Algorithm 4.9 is $O(m^2 n^2)$.

Because $\Sigma_o \subseteq \Sigma$, $n = |\Sigma|$ in this section is greater than $n = |\Sigma_o|$ in Section III-B.

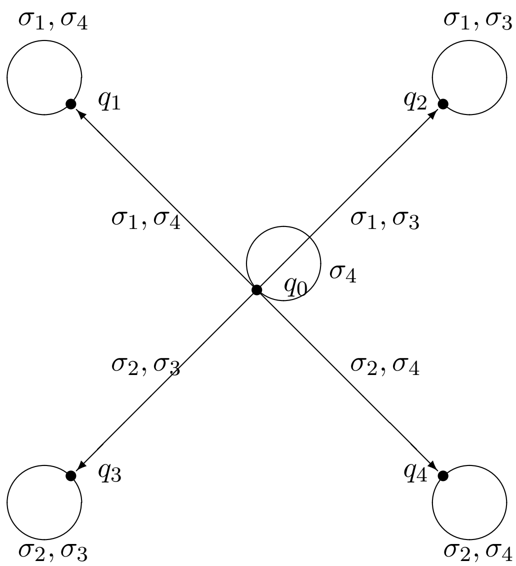

Fig. 7. An Automata.

V. EXAMPLES

A. *Example of Algorithm in Section III-B*

Let us consider the system which is visualized as Fig. 7:

From Fig. 7, we know $Q = \{q_0, q_1, q_2, q_3, q_4\}$ and $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. It is easy to compute that: $Q(\sigma_1) = \{q_1, q_2\}$, $Q(\sigma_2) = \{q_3, q_4\}$, $Q(\sigma_3) = \{q_2, q_3\}$, $Q(\sigma_4) = \{q_0, q_1, q_4\}$.

Diagnosability of the circuit depends on Σ_o and T. Let the desired partition $T = \{\{q_0\}, \{q_1\}, \{q_3\}\}$. We consider the following two examples for Σ_o.

Let $\Sigma_o = \{\sigma_1, \sigma_2\}$, we first use Algorithm 3.3 to compute $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$. In "Initialization" section, set $m = 5, n = 2, s_0 = s_1 = s_2 = s_3 = s_4 = 0$. In "Intersection"recycle, when step $i = 1(\sigma_1)$, we get $s_1 = s_2 = 1$; when step $i = 2(\sigma_2)$, we get $s_3 = s_4 = 2$. The final result is $s_0 = 0, s_1 = s_2 = 1, s_3 = s_4 = 2$. And then we send the result to Algorithm 3.4. By using quicksort algorithm, we get the result $s_0 < s_1 = s_2 < s_3 = s_4$. In the last statements of Algorithm 3.4, we find $s_1 = s_2$. But in the desired partition T, $q_1 \in \{q_1\}$, and $q_2 \overline{\in} \{q_1\}$, so we get the $OFD = FALSE$ in final. So the system is not diagnosable with respect to T and Σ_o.

Let $\Sigma_o = \{\sigma_1, \sigma_2, \sigma_3\}$, we first use Algorithm 3.3 to compute $\bigwedge\limits_{\sigma \in \Sigma_o} T_\sigma$. In "Initialization" section, set $m = 5, n = 3, s_0 = s_1 = s_2 = s_3 = s_4 = 0$. In "Intersection"recycle, when step $i = 1(\sigma_1)$, we get $s_1 = s_2 = 1$; when step $i = 2(\sigma_2)$, we get $s_3 = s_4 = 2$; when step $i = 3(\sigma_3)$, we get $s_2 = 5, s_3 = 6$. The final result is $s_0 = 0, s_1 = 1, s_2 = 5, s_3 = 6, s_4 = 2$. And then we send the result to Algorithm 3.4. By using quicksort algorithm, we get the result $s_0 < s_1 < s_4 < s_2 < s_3$. In the last statements of Algorithm 3.4, all

the values of $s_j(j=0,1,2,3,4)$ are not equal, and we get the $OFD = TRUE$ in final. So the system is diagnosable with respect to T and Σ_o.

B. Lost of Algorithm 2.1

Let

$$M = \{\sigma_1, \sigma_2, \ldots, \sigma_{10}\}.$$

And the cost of M is

$$C(M) = \{27, 23, 20, 15, 10, 9, 7, 5, 4, 1\},$$

G is diagnosable with respect to given T and an element of $OES(T)$.

$OES(T) = \{E_1 = \{\sigma_3, \sigma_5, \sigma_7, \sigma_{10}\}\} \cup \{E_2 = \{\sigma_3, \sigma_5, \sigma_8, \sigma_9, \sigma_{10}\}\} \cup \{\Sigma \subseteq M : E_1 \subseteq \Sigma, or, E_2 \subseteq \Sigma\}$. Using Algorithm 2.1, we can get the $minOES = E_2$, but the cost of E_2 is $20+10+5+4+1 = 40$. The cost of E_1 is $20+10+7+1 = 38$, so the $minOES$ is not the minimal cost of $OES(T)$.

C. Example of Algorithm 4.3

Using Algorithm 4.3, in "Initialization" section, we get $minOES = \{\sigma_3, \sigma_5, \sigma_8, \sigma_9, \sigma_{10}\}$, $lmS = minL(minOES) = 3$, $cmS = C(minOES) = 40$. And then we get

$$H = \left\{ \begin{array}{l} \{\sigma_1, \sigma_5, \sigma_{10}\}, \\ \{\sigma_1, \sigma_6, \sigma_9\}, \\ \{\sigma_1, \sigma_6, \sigma_{10}\}, \\ \{\sigma_1, \sigma_7, \sigma_8, \sigma_{10}\}, \\ \{\sigma_1, \sigma_7, \sigma_9, \sigma_{10}\}, \\ \{\sigma_1, \sigma_8, \sigma_9, \sigma_{10}\}, \\ \{\sigma_2, \sigma_4, \sigma_{10}\}, \\ \{\sigma_2, \sigma_5, \sigma_7\}, \\ \{\sigma_2, \sigma_5, \sigma_8, \sigma_{10}\}, \\ \{\sigma_2, \sigma_5, \sigma_9, \sigma_{10}\}, \\ \{\sigma_2, \sigma_6, \sigma_7, \sigma_{10}\}, \\ \{\sigma_2, \sigma_6, \sigma_8, \sigma_{10}\}, \\ \{\sigma_2, \sigma_6, \sigma_9, \sigma_{10}\}, \\ \{\sigma_2, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, \\ \{\sigma_3, \sigma_4, \sigma_8\}, \\ \{\sigma_3, \sigma_4, \sigma_9, \sigma_{10}\}, \\ \{\sigma_3, \sigma_5, \sigma_6, \sigma_{10}\}, \\ \{\sigma_3, \sigma_5, \sigma_7, \sigma_{10}\}, \\ \{\sigma_3, \sigma_5, \sigma_8, \sigma_9, \sigma_{10}\}, \\ \{\sigma_3, \sigma_6, \sigma_7, \sigma_9\}, \\ \{\sigma_3, \sigma_6, \sigma_7, \sigma_{10}\}, \\ \{\sigma_3, \sigma_6, \sigma_8, \sigma_9, \sigma_{10}\}, \\ \{\sigma_3, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, \\ \textit{other not empty subset of above set.} \end{array} \right\}$$

In "Testing diagnosability" recycle, we find that only two elements of H(E_1 and E_2) are diagnosable, and $C(E_1) < C(E_2)$. Therefor we get the minimal cost of $OES(T)$ is $C(E_1)$.

If we consider the set NES. From Algorithm 4.9, we get the set NES. We partition the finite events set Σ into disjoint subsets $NES = \{\sigma_3, \sigma_5, \sigma_{10}\}$ and $\Sigma \setminus NES = \{\sigma_1, \sigma_2, \sigma_4, \sigma_6, \sigma_7, \sigma_8, \sigma_9\}$. Now in Algorithm 4.3, we use the set $(\Sigma \setminus NES)$ as the set Σ. The computing procedure is as follows: in "Initialization"

section, we get $minOES = \{\sigma_8, \sigma_9\}$, $lmS = minL(minOES) = 8$, $cmS = C(minOES) = 9$. And then we get that

$$H = \left\{ \begin{array}{l} \{\sigma_6\}, \\ \{\sigma_7\}, \\ \{\sigma_8, \sigma_9\}, \\ \textit{other not empty subset of above set.} \end{array} \right\}$$

In "Testing diagnosability" recycle, we find that only two elements of H($\{\sigma_7\}$ and $\{\sigma_8, \sigma_9\}$) are diagnosable, and $C(\{\sigma_7\}) < C(\{\sigma_8, \sigma_9\})$. Hence we get that the minimal cost of $OES(T)$ is $C(\{\sigma_7\}) + C(NES)$. The result is the same as that by using the method above, but the complexity is greatly reduced.

VI. Conclusion

In terms of some problems in off-line diagnostics [15], in this paper, we present some off-line diagnostic algorithms to overcome the shortcomings. We give a general method of judging a system's off-line diagnosability, which is a polynomial-time algorithm. And we give an algorithm of how to find the minimal set in all observable event sets. Of course, another issue worthy of further consideration is the on-line diagnostic algorithms of the minimal cost in DESs. We would like to consider it in subsequent work.

References

[1] S. Bavshi and E. Chong, "Automated fault diagnosis of using a discrete event systems framework," in *Proc. 9th IEEE int. Symp. Intelligent Contr.*, 1994, pp. 213-218.

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Boston, MA: Kluwer, 1999.

[3] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Automat. Contr.*, vol. 33, no. 3, pp. 249-260, Mar. 1988.

[4] A. Darwiche and G. Provan, "Exploiting system structure in modelbased diagnosis of discrete event systems," in *Proc. 7th Annu. Int. Workshop on the Principles of Diagnosis (DX'96)*, Oct. 1996, pp. 95-105.

[5] R. David and H. Alla, *Petri Nets and Grafcet: Tools for modelling discrete event systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[6] R. Debouk, "Failure diagnosis of decentralized discrete event systems," Ph.D. dissertation, Elec. Eng. Comp. Sci. Dept., University of Michigan, Ann Arbor, MI, 2000.

[7] R. Debouk, S. Lafortune, and D. Teneketzis,"On the effect of communication delays in failure diagnosis of decentralized discrete event systems," *Discrete Event Dyna. Syst.: Theory Appl.*, 13(2003), pp. 263-289.

[8] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge based redundancy-A survey and some new results," *Automatica*, vol. 26, pp. 459-474, 1990.

[9] E. Garcia, F. Morant, R. Blasco-Giminez, A. Correcher, and E. Quiles, "Centralized modular diagnosis and the phenomenon of coupling," in *Proc. 2002 IEEE Int. Workshop on Discrete Event Systems (WODES'02)*, Oct. 2002, pp. 161-168.

[10] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addision-Wesley, 1979.

[11] S. Jiang and R. Kumar, "Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications," in *Proc. 2002 Amer. Control Conf.*, May 2002, pp. 128-133.

[12] S. Jiang, R. Kumar, and H. Garcia, "Diagnosis of repeated failures in discrete event systems," in *Proc. 41st IEEE Conf. Decision and Control*, Dec. 2002, pp. 4000-4005.

[13] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen, "Failure diagnosis of dynamic systems: An approach based on discrete-event systems," in *Proc.2001 Amer. Control Conf.*, Jun. 2001, pp. 2058-2071.

[14] G. Lamperti and M. Zanella, "Diagnosis of discrete event systems integrating synchronous and asynchronous behavior," in *Proc. 9th Int. Workshop on Principles of Diagnosis (DX'99)*, 1999, pp. 129-139.

[15] F. Lin, Diagnosability of Discrete Event Systems and Its Applications, Disctete Event Dynamic Systems, 4(197-212), 1994.

[16] F. Lin, Zhenghui Lin, T.William Lin, A Uniform Approach to Mixed-signal Circuit Test, International Journal of Circuit Theory and Applications, 25(81-93), 1997.

[17] J. Lunze and J. Schröder, "State observation and Diagnosis of discrete-event systems described by stochastic automata," *Discrete Event Dyna. Syst.: Theory Appl.*, vol. 11, pp. 319-369, 2001.

[18] D. Pandalai and L. Holloway, "Template languages for fault monitoring of discrete event processes," *IEEE Trans. Automat. Contr.*, vol. 45, no. 5, pp. 868-882, May 2000.

[19] Y. Pencolé, "Decentralized diagnoser approach: Application to telecommunication networks," in *Proc. 11th Int. Workshop on Principles of Diagnosis (DX'00)*, Jun. 2000, pp. 185-192.

[20] G. Provan and Y.-L. Chen, "Diagnosis of timed discrete event systems using temporal causal networks: Modeling and analysis," in *Proc. 1998 Int. Workshop on Discrete Event Systems (WODES'98)*, Aug. 1998, pp. 152-154.

[21] G. Provan and Y.-L. Chen, "Model-based diagnosis and control reconfiguration for discrete event systems: An integrated approach," in *Proc. 38th IEEE Conf. Decision and Control*, Dec. 1999, pp. 1762-1768.

[22] L. Rozé and M. O. Cordier,"Diagnosing discrete event systems: Extending the "Diagnoser Approach" to deal with telecommunication networks," *Discrete Event Dyna. Syst.: Theory Appl.*, vol. 12, pp. 43-81, 2002.

[23] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 7, pp. 908-929, Jul. 1998.

[24] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 40, no. 9, pp. 1555-1575, Sep. 1995.

[25] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Trans. Automat. Contr. Syst. Technol.*, vol. 4, no. 2, pp. 105-124, Mar. 1996.

[26] R. Sengupta, "Discrete-event diagnostics of automated vehicles and highways," in *Proc. 2001 Amer. Control Conf.*, Jun. 2001.

[27] K. Sinnamohideen, "Discrete-event diagnostics of heating, ventilation, and air-conditioning systems," in *Proc. 2001 Amer. Control Conf.*, Jun. 2001, pp. 2072-2076.

[28] R. Su and W. Wonham, "Global and local consistencies in distributed fault diagnosis for discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 50, no. 12, pp. 1923-1935, Dec. 2005.

[29] D. Thorsley, and D. Teneketzis, "Diagnosability of stochastic discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 50, no. 4, pp. 476-492, April. 2005.

[30] N. Viswanadham and T. Johnson, "Fault detection and diagnosis of automated manufacturing systems," in *Proc. 27th IEEE Conf. Decision and Control*, Dec. 1988, pp. 2301-2306.

[31] G. Westerman, R. Kumar, C. Stround, and J. Heath, "Discrete event system approach for delay fault analysis in digital circuits," in *Proc. 1998 Amer. Control Conf.*, Jun. 1998, pp. 239-243.

[32] S. H. Zad, R. Kwong, and W. Wonham, "Fault diagnosis in discrete event systems: Framework and model reduction," in *Proc. 37th IEEE Conf. Decision and Control*, Dec. 1998, pp. 3769-3774.